BOOKSTORE TEST CASE REPORT

I built a RESTful API for a fictional online bookstore using Java and JAX-RS (with Jersey). I stored all the data in-memory using Java collections, like HashMap and ArrayList. This allowed me to keep the API simple and fully focused on logic and REST principles, as required by the brief. Lastly, to test everything, I used Postman to send requests and verify that all endpoints worked correctly. The API runs on Grizzly HTTP server, making it easy to deploy and test locally.

The Custom Exceptions I used:

* AuthorNotFoundException - I used this when a client tries to view, update, or delete an author that doesn’t exist in the system.  
  For example, In the demo video I used /api/authors/2000 and that ID isn’t in memory, this exception is triggered to let them know the author isn’t there and doesn’t exist hasn’t been made.
* BookNotFoundException - This one is thrown when someone tries to access or work with a book ID that doesn't exist — like adding it to a cart or deleting a book that's not in the system. It's especially helpful in the cart and order flow to make sure users don’t order books that aren’t listed.
* CustomerNotFoundException - I used this to handle any situation where a customer ID doesn’t match any existing customer in my data store.  
  This could happen when trying to view a cart or place an order using an ID that was never registered.
* CartNotFoundException - This is used if someone tries to view or update a cart for a customer that doesn’t have one yet.  
  It prevents errors and makes it clear to the user that they need to add items before accessing the cart.
* InvalidInputException - I created this for handling bad or missing data in requests — like trying to create a customer without entering a name or email. This keeps my API clean and avoids saving incomplete or broken entries.
* Lastly, OutOfStockException - This exception is thrown when the customer tries to order more books than are currently in stock.  
  It helps stop the order from going through unless the quantity is valid, which is realistic for how online stores work.

All My Test Cases

Author Resource Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Endpoint | Description | Method | Request Body (JSON) | Expected Http Status Code | Expected Response Bosy (JSON) | ACTUAL RESULTS |
| /api/authors | This is to create a new author in the system. | POST | {      "authorId": 1,      "authorNationality":"French",      "authorName": "Giroud lacazette",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | It should work and show a 201 created to show it was successful. | "authorId": 1,      "authorNationality":"French",      "authorName": "Giroud lacazette",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | "authorId": 1,      "authorNationality":"French",      "authorName": "Giroud lacazette",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  (WORKED with a 201 message) |
| /api/authors/2000 | This is an example of trying to get a non-existing author at author id 2000. | GET | N/A | Should show a 404 NOT FOUND | {“error": "Author Not Found", “message”: “Opps, Sorry the Author with this ID was not found} | {“error": "Author Not Found", “message”: “Opps, Sorry the Author with this ID was not found}    (WORKED) |
| /api/authors | This is getting all the authors into the system. | GET | N/A | Should show a 200 Ok. | {      "authorId": 1,      "authorNationality":"French",      "authorName": "Giroud lacazette",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  {      "authorId": 2,      "authorNationality":"French",      "authorName": "John",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | {      "authorId": 1,      "authorNationality":"French",      "authorName": "Giroud lacazette",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }      "authorId": 2,      "authorNationality":"French",      "authorName": "John",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  (WORKED it showed the two authors I created in the demo as well) |
| /api/authors/1 | This is getting a specific author at a specific ID. | GET | N/A | It will show with a 200 message to show that it has been brought. | {      "authorId": 1,      "authorNationality":"French",      "authorName": " Giroud lacazette ",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | {      "authorId": 1,      "authorNationality":"French",      "authorName": " Giroud lacazette ",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  (WORKED – it brings this specific author based on it’s ID) |
| /api/authors/1 | This is specifically deleting the author with the ID:1. | DELETE | N/A | 204 No Content Message should show I delete the author with the author id of 1, so only the author 2 should remain. | "authorId": 2,      "authorNationality":"French",      "authorName": "John",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | "authorId": 2,      "authorNationality":"French",      "authorName": "John",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  (WORKED) Giroud was deleted and now john only remains. |
| /api/authors/1 | PUT |  | {      "authorId": 1,      "authorNationality":"French",      "authorName": "djdjd",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | It should show a 200 ok message to show it’s been updated. | {      "authorId": 1,      "authorNationality":"French",      "authorName": "djdjd",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  } | {      "authorId": 1,      "authorNationality":"French",      "authorName": "djdjd",      "authorBio": "French  author of 1850s",      "authorGenre": "Thriller",      "authorawards": "Nobel Prize in Literature 1853"  }  (WORKED) as the name was changed as example updated) |

Book Resource Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Endpoint | Description | Method | Request Body (JSON) | Expected Http Status Code | Expected Response Bosy (JSON) | ACTUAL RESULTS |
| /api/books | Creating a book into the system. | POST | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | Should show a 201 message to show it’s been created successfully. | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09  (WORKED) With a 201 message as well so it worked, and this book was created successfully) |
| /api/books | Creating a book with a | POST | "bookId": 1, "authorId": 8000, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | Expected to show a 404 Not Found - AuthorNotFoundException | 404 Not Found – AuthorNotFoundException  With a message (“Opps, Sorry the Author with this ID was not found) | 404 Not Found – AuthorNotFoundException (“Message” “Opps, Sorry the Author with this ID was not found)  (WORKED) This is due to the fact an author with this ID hasn’t been created yet. |
| /api/books | This is to view and get all the books in the system | GET | N/A | 200 OK with list of books | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09  (WORKED) |
| /api/books/1 | This is to get a specific book at the book id 1. | GET | N/A | |  | | --- | |  |  |  | | --- | | 200 OK with book JSON | | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09  200 OK message with book JSON at that specific ID. | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09  (WORKED)  200 OK message with book JSON at that specific ID. |
| /api/books/400 | This is to get a specific book at this ID, but it’s invalid as it doesn’t exist. | GET | N/A | 404 Not Found BookNotFoundException | 404 Not Found BookNotFoundException    (With a message saying “Opps, sorry the book with ID 400 was not found. | 404 Not Found BookNotFoundException    (With a message saying “Opps, sorry the book with ID 400 was not found.  (WORKED) |
| /api/books/1 | This is to update the Book at the ID 1 So what I changed was book title to stars. | PUT | "bookId": 1, "authorId": 1, "bookTitle: "stars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | 200 OK with the JSON. Showing its been updated. | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 | "bookId": 1, "authorId": 1, "bookTitle": "Adventures in Mars", "bookIsbn": "9780451524935", "publicationYear": 1950, "bookLanguage": "English", "bookPrice": 13.99, "bookStock": 109, "bookRating": 4.09 |
| /api/books/1 | Deleting the book at the ID:1. | DELETE | N/A | 204 No Content Message. To show it’s been deleted. | It should have deleted it and kept on the book at ID 2. | It only showed the Book at ID 2 not the deleted book showing the deletion worked.  (WORKED). |

Customer Resource Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Endpoint | Description | Method | Request Body (JSON) | Expected Http Status Code | Expected Response Bosy (JSON) | ACTUAL RESULTS |
| /api/customers | Creating a customer in the system. | POST | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" } | 201 Created + customer JSON | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" } | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" }  (Worked)  It showed a 201 created and a message. |
| /api/customers | This will get all the customers in the system. | GET | N/A | 200 OK. | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" } | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" }  (WORKED) |
| /api/customers/1 | This will get the specific customer at ID 1. | GET | N/A | 200 OK. | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" } | { "customerId": 1, "customerName": "Hamza Hassan", "gender": "Male", "customerEmail": "hamza@example.com", "customerPassword": "UniversityofW", "customerPhoneNumber": "075425256172", "billingAddress": "39 Cowan court nw10 87a", "shippingAddress": "39 Cowan court nw10 87a" }  (WORKED) |
| /api/customers/1 | This is to update the customer information. Here I updated my name. | PUT | { "customerName": "John", "customerEmail": "hamzah@example.com" } | 200 OK + updated JSON | { "customerName": "John", "customerEmail": "hamzah@example.com" } | { "customerName": "John", "customerEmail": "hamzah@example.com" }  (Worked) |
| /api/customers/1 | This will delete the customer with the Customer ID 1. | DELETE | N/A | 204 No Content Message. To show it’s been deleted. | It should only show the Customer 2 I made and should delete the Customer with the ID 1. | (Worked) |

Cart Resource Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Endpoint | Description | Method | Request Body (JSON) | Expected Http Status Code | Expected Response Bosy (JSON) | ACTUAL RESULTS |
| /api/customers/1/cart/items | This will add a item to the cart. | POST | |  | | --- | |  |  |  | | --- | | { "bookId": 1, "bookCount": 3, "bookTitle": "Alice in Wonderland", "bookPrice": 38.99, "cartSubTotal": 50.98} | | 201 Created + item added | |  | | --- | |  |  |  | | --- | | { "bookId": 1, "bookCount": 3, "bookTitle": "Alice in Wonderland", "bookPrice": 38.99, "cartSubTotal": 50.98 } | | |  | | --- | |  |  |  | | --- | | { "bookId": 1, "bookCount": 3, "bookTitle": "Alice in Wonderland", "bookPrice": 38.99, "cartSubTotal": 50.98 } |   (WORKED) |
| /api/customers/1/cart | This will get the cart based on customer ID 1. | GET | N/A | 200 OK + list of cart items | {  "bookId": 1,  "bookCount": 3,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  {  "bookId": 2,  "bookCount": 20,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  } | {  "bookId": 1,  "bookCount": 3,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  {  "bookId": 2,  "bookCount": 20,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  (WORKED) |
| /api/customers/900/cart | This is getting the cart for a non-existing customer ID: 900. | GET | N/A | |  | | --- | |  |  |  | | --- | | 404 Not Found -CustomerNotFoundException | | |  | | --- | |  |  |  | | --- | | 404 Not Found - CustomerNotFoundException | | |  | | --- | |  |  |  | | --- | | 404 Not Found - CustomerNotFoundException |   (WORKED) |
| /api/customers/1/cart/items/1 | This is to update the cart item quantity for the customer ID 1. | PUT | { "bookCount": 5} | 200 OK + the updated book count. | {  "bookId": 1,  "bookCount": 5,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  {  "bookId": 2,  "bookCount": 20,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  } | {  "bookId": 1,  "bookCount": 5,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  {  "bookId": 2,  "bookCount": 20,  "bookTitle": "Alice in Wonderland",  "bookPrice": 38.99,  "cartSubTotal": 50.98  }  (WORKED |
| /api/customers/1/cart/items/999 | Deleting the cart item with a non-existed CustomerID. | DELETE | N/A | 404 Not Found - CartNotFoundException | 404 Not Found - CartNotFoundException | 404 Not Found – CartNotFoundException  (WORKED) |

Order Resource Test Cases:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Endpoint | Description | Method | Request Body (JSON) | Expected Http Status Code | Expected Response Bosy (JSON) | ACTUAL RESULTS |
| /api/customers/1/orders | Place an order with valid cart | POST | N/A | 201 Created with full order JSON: | |  | | --- | |  |  |  | | --- | | 201 Created with full order JSON: { "orderId": 1, "customerId": 1, "orderedItems": [...], "dateOrdered": <timestamp> } | | |  | | --- | |  |  |  | | --- | | 201 Created with full order JSON: { "orderId": 1, "customerId": 1, "orderedItems": [...], "dateOrdered": <timestamp> }  (WORKED) | |
| /api/customers/1/orders | Get all orders for a customer | GET | N/A | 200 OK – shwoing it has been brought and its successful. | |  | | --- | |  |  |  | | --- | | 200 OK with list of orders for customer 1 | | |  | | --- | |  |  |  | | --- | | 200 OK with list of orders for customer 1 |   (WORKED) |
| /api/customers/1/orders/999 |  | GET | N/A | 404 Not Found | |  | | --- | |  |  |  | | --- | | 404 Not Found with message: { "error": "InvalidInputException", With a message saying ("Sorry, but there are no orders that were found for this customer."); | | |  | | --- | |  |  |  | | --- | | 404 Not Found with message: { "error": "InvalidInputException", With a message saying ("Sorry, but there are no orders that were found for this customer."); |   (WORKED) |

Just to conclude, all endpoints were tested using Postman, including both valid and invalid inputs.  
Each test was checked for correct HTTP status codes and proper error handling using my custom exceptions. In the demo video, I focused on and explained the following exceptions: AuthorNotFoundException, BookNotFoundException, CustomerNotFoundException, and CartNotFoundException. While InvalidInputException and OutOfStockException were implemented in the code and tested separately, I did not demonstrate them in the video due to time. Overall, this report confirms that the API is fully functional, follows REST principles, and meets the coursework brief.